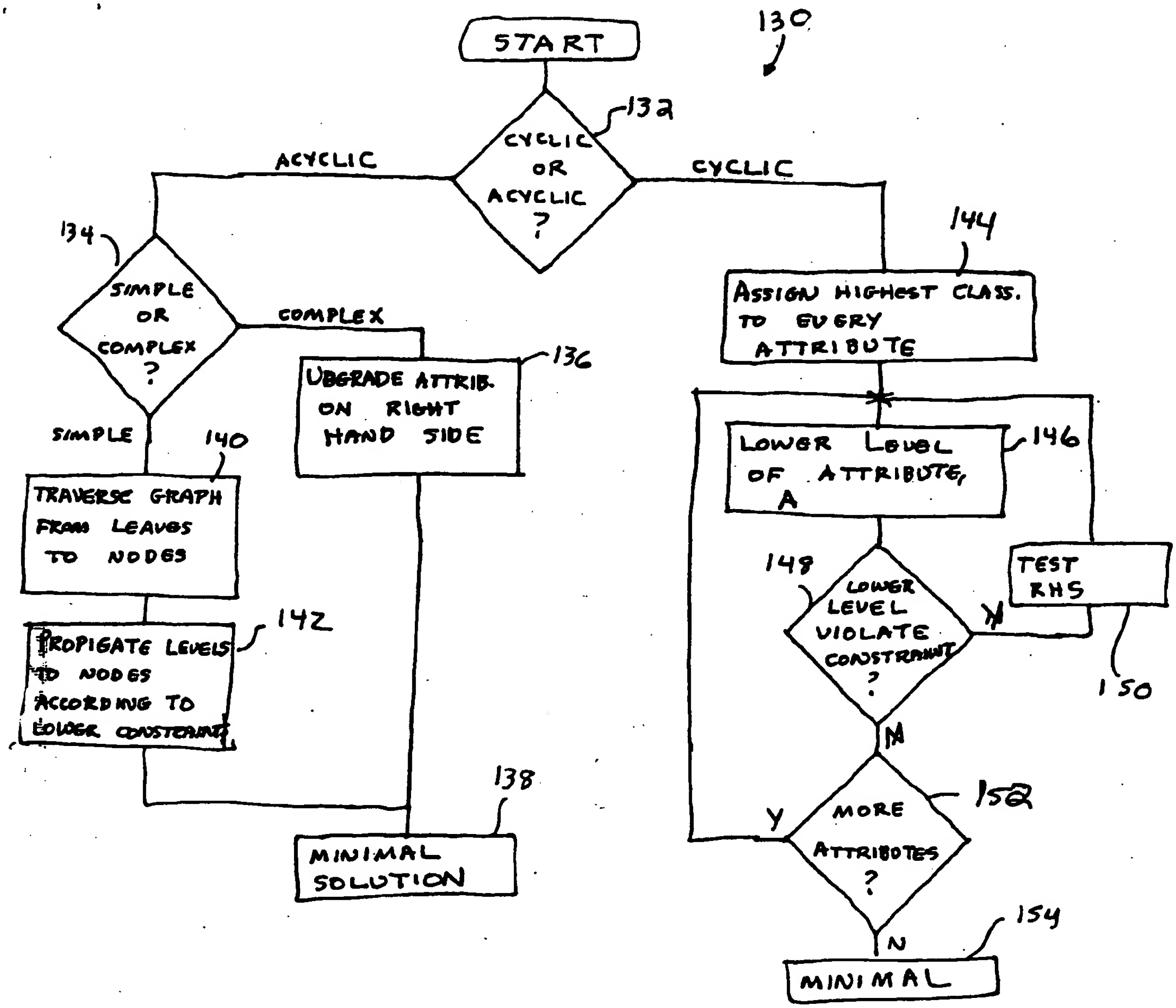
130
START
132
CYCLIC OR ACYCLIC ?
ACYCLIC
CYCLIC
134
SIMPLE OR COMPLEX ?
COMPLEX
SIMPLE
136
UBGRADE ATTRIB. ON RIGHT HAND SIDE
140
TRAVERSE GRAPH FROM LEAVES TO NODES
142
PROPIGATE LEVELS TO NODES ACCORDING TO LOWER CONSTRAINTS
138
MINIMAL SOLUTION
144
ASSIGN HIGHEST CLASS. TO EVERY ATTRIBUTE
146
LOWER LEVEL OF ATTRIBUTE, A
148
LOWER LEVEL VIOLATE CONSTRAINT ?
Y
TEST RHS
150
N
152
MORE ATTRIBUTES ?
Y
N
154
MINIMAL

FIGURE 6

$\lambda(\texttt{illness}) \succeq \text{Research}$
$\lambda(\texttt{prescription}) \succeq \text{Clinical}$
$\lambda(\texttt{prescription}) \succeq \lambda(\texttt{treatment})$
$\lambda(\texttt{treatment}) \succeq \text{Public}$
$\lambda(\texttt{treatment}) \succeq \lambda(\texttt{visit})$
$\lambda(\texttt{treatment}) \succeq \lambda(\texttt{illness})$
$\lambda(\texttt{visit}) \succeq \text{Public}$

Figure 7 (a)

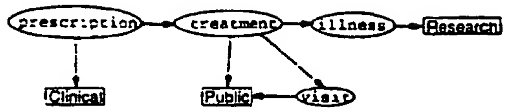

Figure 7 (b)

$\lambda(\texttt{illness}) = \text{Research}$
$\lambda(\texttt{prescription}) = \text{Clinical}$
$\lambda(\texttt{treatment}) = \text{Research}$
$\lambda(\texttt{visit}) = \text{Public}$

Figure 7 (c)

$\lambda(\texttt{patient}) \succeq \text{Public}$
$\lambda(\texttt{bill}) \succeq \text{Financial}$
$\text{lub}\{\lambda(\texttt{patient}), \lambda(\texttt{bill})\} \succeq \text{Admin}$

Figure 8 (a)

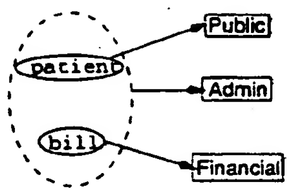

Figure 8 (b)

$\lambda(\texttt{bill}) = \text{Admin}$
$\lambda(\texttt{patient}) = \text{Public}$

Figure 8 (c)

$\lambda(\texttt{bill}) = \text{Financial}$
$\lambda(\texttt{patient}) = \text{Research}$

Figure 8 (d)

$\lambda(\texttt{division}) \succeq \text{Public}$
$\text{lub}\{\lambda(\texttt{division}), \lambda(\texttt{plan})\} \succeq \lambda(\texttt{doctor})$
$\lambda(\texttt{doctor}) \succeq \text{Public}$
$\lambda(\texttt{doctor}) \succeq \lambda(\texttt{illness})$
$\lambda(\texttt{illness}) \succeq \lambda(\texttt{division})$
$\lambda(\texttt{illness}) \succeq \text{Research}$
$\lambda(\texttt{plan}) \succeq \text{Financial}$

Figure 9 (a)

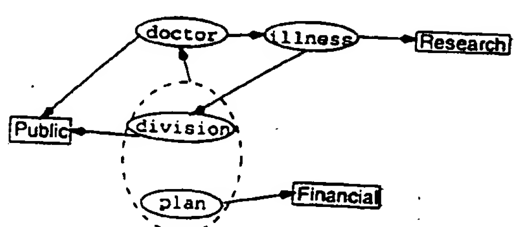

Figure 9 (b)

$\lambda(\texttt{division}) = \text{Public}$
$\lambda(\texttt{doctor}) = \text{Research}$
$\lambda(\texttt{illness}) = \text{Research}$
$\lambda(\texttt{plan}) = \text{Admin}$

Figure 9 (c)

Algorithm 3.1 (Minimal Classification Generation)

```
MAIN
For A ∈ 𝒜 do Constr[A] := ∅; visit[A] := 0; done[A] := FALSE
For l ∈ L do done[l] := TRUE; visit[l] := 1
For c=(lhs,rhs) ∈ C_lower do
   count[c] := 0
   For A ∈ lhs do
      Constr[A] := Constr[A] ∪ {c}; count[c] := count[c] + 1
Stack := ∅
For A ∈ 𝒜 do
   If visit[A] = 0 then dfs_visit(A)
max_scc := 0
For i = 1,...,|𝒜| do scc[i] := ⟨⟩
For A ∈ 𝒜 do visit[A] := 0
While NOTEMPTY(Stack) do
     A := POP(Stack)
     If visit[A] = 0 then
        max_scc := max_scc + 1
        scc[max_scc] := ⟨A⟩
        dfs_back_visit(A)
For A ∈ 𝒜 do λ(A) := ⊤; visit[A] := 0
compute_upper_bounds
compute_partial_lubs
compute_minimal_solution

COMPUTE_UPPER_BOUNDS
For (l, A) ∈ C_upper do λ(A) := λ(A) ⊓ l
For i := 1,..., max_scc do
   For A ∈ scc[i] do
      If visit[A] = 0 then upper_bound(A, i)

UPPER_BOUND(A, i)
visit[A] := 1
For c = (lhs, rhs) ∈ Constr[A] do
   If count[c] > 0 then count[c] := count[c] − 1
   If count[c] = 0 or rhs ∈ scc[i] then
      levlhs := ⊥
      For A' ∈ lhs do levlhs := levlhs ⊔ λ(A')
      If ¬(levlhs ⪰ λ(rhs)) then
        If rhs ∈ L then Fail
        else λ(rhs) := λ(rhs) ⊓ levlhs
              If rhs ∈ scc[i] then
                 upper_bound(rhs, i)

COMPUTE_MINIMAL_SOLUTION
For i := max_scc,..., 1 do
   For A ∈ scc[i] do
      done[A] := TRUE; l := ⊥
      For c=(lhs,rhs) ∈ Constr[A] do
         If done[rhs] then
            case |lhs| of
               1:  l := l ⊔ λ(rhs)
               >1: l := l ⊔ minlevel(A, c)
         else done[A] := FALSE
      If done[A] then λ(A) := l
      else DSet := {l' | l' is a maximal level, λ(A)≻l' ⪰ l}
           While DSet ≠ ∅
              Choose l'' in DSet; DSet := DSet − l''
              Lower := try_to_lower(A, l'')
              If Lower ≠ ∅ then
                 For (A', l') ∈ Lower do λ(A') := l'
                 DSet := {l' | l' maximal level, λ(A)≻l' ⪰ l}
           done[A] := TRUE
      For c ∈ Constr[A] do
         j := count[c]; Plub[c][j] := λ(A) ⊔ Plub[c][j + 1]
         count[c] := count[c] − 1

DFS_VISIT(A)
visit[A] := 1
For (lhs, rhs) ∈ Constr[A] do
   If visit[rhs] = 0 then dfs_visit(rhs)
PUSH(A, Stack)

DFS_BACK_VISIT(A)
/* Traverses the constraints backward and inserts all
attributes found in the same SCC list as A */
visit[A] := 1
For (lhs, A) ∈ C_lower do
   For A' ∈ lhs do
      If visit[A'] = 0 then
         scc[max_scc] := concat(⟨A'⟩, scc[max_scc])
         dfs_back_visit(A')

COMPUTE_PARTIAL_LUBS
For c=(lhs,rhs) ∈ C_lower do count[c] := 0; Plub[c][0] := ⊥
For i := 1,..., max_scc do
   For A ∈ reverse(scc[i]) do
      For c = (lhs, rhs) ∈ Constr[A] do
         count[c] := count[c] + 1; j := count[c]
         Plub[c][j] := Plub[c][j − 1] ⊔ λ(A)
For c=(lhs,rhs) ∈ C_lower do j := count[c] + 1; Plub[c][j] := ⊥

MINLEVEL(A,c)
/* Returns a minimal level for A that keeps c satisfied */
j := count[c]; (lhs, rhs) := c; last := λ(A)
lubothers := Plub[c][j − 1] ⊔ Plub[c][j + 1]
If lubothers ⪰ λ(rhs) then last := ⊥
else Try:={l | l is a maximal level s. t. last≻l}
     While Try ≠ ∅ do
        Choose l in Try; Try := Try − l
        If (l ⊔ lubothers) ⪰ λ(rhs) then
           last := l; Try:={l | l is a maximal level s. t. last≻l}
return last

TRY_TO_LOWER(A,l)
Tocheck := {(A, l)}
Tolower := ∅
Repeat
   Choose (A', l') ∈ Tocheck
   Tocheck := Tocheck − {(A', l')}
   Tolower := Tolower ∪ {(A', l')}
   For (lhs, rhs) ∈ Constr[A'] do
      level := ⊥
      For A'' ∈ lhs do
         If ∃(A'', l'') ∈ Tolower then
            level := level ⊔ l''
         else level := level ⊔ λ(A'')
      case done[rhs] of
         TRUE: If ¬(level ⪰ λ(rhs)) then return ∅
         FALSE: If ¬(level ⪰ λ(rhs)) then
                  newlevel := λ(rhs) ⊓ level
                  If ∃(rhs, l'') ∈ (Tolower ∪ Tocheck) then
                     If ¬(newlevel ⪰ l'') then
                        newlevel := l'' ⊓ newlevel
                        If (rhs, l'') ∈ Tolower then
                           Tolower := Tolower − {(rhs, l'')}
                        else Tocheck := Tocheck − {(rhs, l'')}
                        Tocheck := Tocheck ∪ {(rhs, newlevel)}
                  else Tocheck := Tocheck ∪ {(rhs, newlevel)}
until Tocheck = ∅
return Tolower
```

Figure 10 Algorithm for computing a minimal classification.

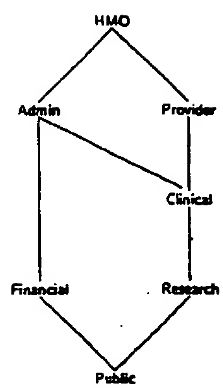

Figure 11 (a)

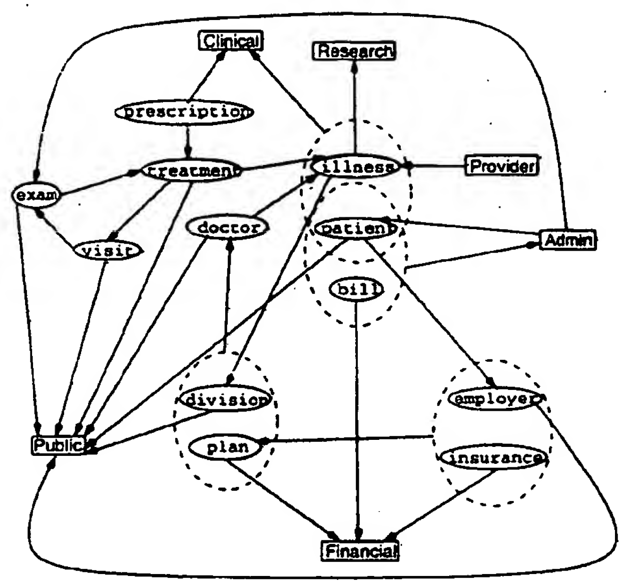

Figure 11 (b)

		SCC [8]			[7]	[6]	[5]	[4]	[3]	[2]			[1]
		doctor	division	illness	plan	employer	patient	bill	insurance	exam	treatment	visit	prescription
	• initial levels	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO
	compute_upper_bounds	HMO	Clinical	Clinical	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
doctor	try_to_lower(doctor,Admin)	Admin	Clinical	Clinical	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
	try_to_lower(doctor,Financial) F	Admin	Clinical	Clinical	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
	try_to_lower(doctor,Clinical)	Clinical	Clinical	Clinical	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
	try_to_lower(doctor,Research)	Research	Research	Research	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
	try_to_lower(doctor,Public) F	Research	Research	Research	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
division	–		Public	Research	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
illness	–			Research	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
plan	–				Admin	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
employer	–					Public	Admin	HMO	HMO	Admin	Admin	Admin	HMO
patient	–						Clinical	HMO	HMO	Admin	Admin	Admin	HMO
bill	–							Financial	HMO	Admin	Admin	Admin	HMO
insurance	–								Admin	Admin	Admin	Admin	HMO
exam	try_to_lower(exam,Financial) F									Admin	Admin	Admin	HMO
	try_to_lower(exam,Clinical)									Clinical	Clinical	Clinical	HMO
	try_to_lower(exam,Research)									Research	Research	Research	HMO
	try_to_lower(exam,Public) F									Research	Research	Research	HMO
treatment	–										Research	Research	HMO
visit	–											Research	HMO
prescription	–												Clinical
	• final levels	Research	Public	Research	Admin	Public	Clinical	Financial	Admin	Research	Research	Research	Clinical

Figure 11 (c)

		SCC											
		(8)			(7)	(6)	(5)	(4)	(3)	(2)			(1)
		doctor	division	illness	plan	employer	patient	bill	insurance	exam	treatment	visit	prescription
	• initial levels	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO	HMO
	compute_upper_bounds	HMO	Clinical	Clinical	HMO	Admin	Admin	HMO	HMO	Admin	Admin	Admin	HMO
patient	try_to_lower(patient,Financial)	HMO	Clinical	Clinical	HMO	Financial	Financial	HMO	HMO	Admin	Admin	Admin	HMO
	try_to_lower(patient,Public)	HMO	Clinical	Clinical	HMO	Public	Public	HMO	HMO	Admin	Admin	Admin	HMO
plan	try_to_lower(plan,Admin)	Admin	Clinical	Clinical	Admin	Public	Public	HMO	HMO	Admin	Admin	Admin	HMO
	try_to_lower(plan,Financial)	Admin	Clinical	Clinical	Financial	Public	Public	HMO	HMO	Admin	Admin	Admin	HMO
doctor	try_to_lower(doctor,Financial)F	Admin	Clinical	Clinical	Financial	Public	Public	HMO	HMO	Admin	Admin	Admin	HMO
	try_to_lower(doctor,Clinical)	Clinical	Clinical	Clinical	Financial	Public	Public	HMO	HMO	Admin	Admin	Admin	HMO
	try_to_lower(doctor,Research)F	Clinical	Clinical	Clinical	Financial	Public	Public	HMO	HMO	Admin	Admin	Admin	HMO
division	–	Clinical	Research	Clinical	Financial	Public	Public	HMO	HMO	Admin	Admin	Admin	HMO
illness	–				Financial	Public	Public	HMO	HMO	Admin	Admin	Admin	HMO
employer	–					Public		HMO	HMO	Admin	Admin	Admin	HMO
bill	–							Admin	HMO	Admin	Admin	Admin	HMO
insurance	–								Financial	Admin	Admin	Admin	HMO
exam	try_to_lower(exam,Financial)F									Admin	Admin	Admin	HMO
	try_to_lower(exam,Clinical)									Clinical	Clinical	Clinical	HMO
	try_to_lower(exam,Research)F									Clinical	Clinical	Clinical	HMO
treatment	try_to_lower(treatment,Clinical)										Clinical	Clinical	HMO
visit	–											Clinical	HMO
prescription	–												Clinical
	• final levels	Clinical	Research	Clinical	Financial	Public	Public	Admin	Financial	Clinical	Clinical	Clinical	Clinical

Figure 12

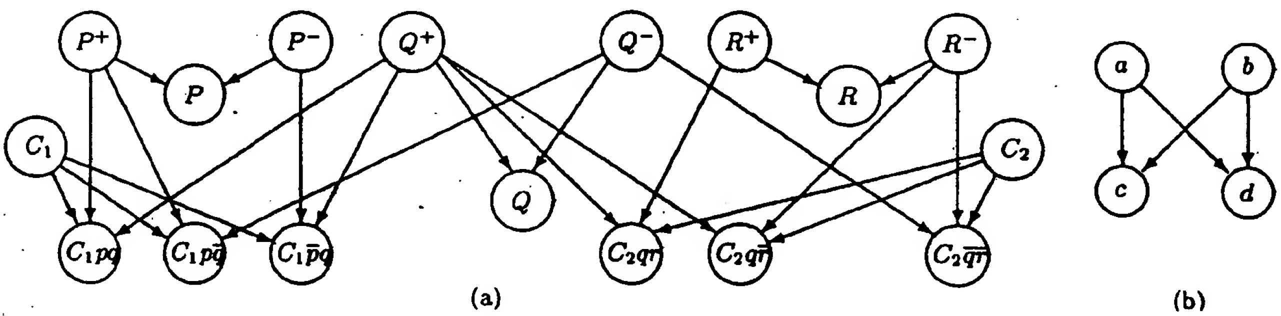

Figure 13